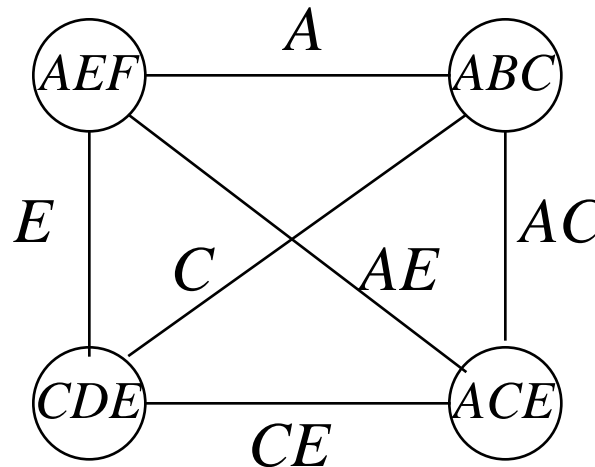


Acyclic constraint networks

- Dual constraint graph of an acyclic constraint network is a tree
 - first characterized in the database literature
 - can apply the tree algorithm to solve acyclic networks
- Dual constraint graph may become a tree by removing redundant edges



Scheduling as constraint satisfaction

- *Scheduling problems*
 - place a set of tasks on a time line
 - subject to temporal constraints, resource constraints, preferences, ...
 - constraints can be *hard* or *soft* and hence...
 - often requires *optimization*, *i.e.*, one tries to maximize the number and type of constraints that are satisfied

Repair-based methods for solving CSPs

- Searches in the space of *complete* assignments
 - can use information about the current assignment that is not available to constructive backtracking
- Particularly useful for solving scheduling problems
 - unexpected events often require dynamic rescheduling
 - scheduling often involves optimization
 - humans appear to find repair-based methods more “natural”

Min-conflicts heuristic for repair

- Motivated by the surprising performance of the GDS neural net
- Min-conflicts repair heuristic
 - Select a variable that is in conflict
 - Assign it a value that minimizes the number of conflicts
- Can be used for both
 - hill-climbing
 - backtracking through the space of complete assignments

Hill climbing with min-conflicts

- Initial assignment created using a greedy algorithm
 - sequentially assign variables to values that minimize conflicts with previously assigned variables
- Repair using the min-conflicts heuristic
 - current assignment may be the min-conflict assignment
- Continue until a solution is found
 - may need to restart

Backtracking using min-conflicts

function *mc-bt(vars-left, vars-done)*

if current assignment is consistent **then return** *true*

 Let *v* be a variable in *vars-left* that is in conflict

 Remove *v* from *vars-left* and add it to *vars-done*

 Let *values* be the values of *v* ordered in ascending order

 according to number of conflicts with variables in *vars-left*

for each *value* in *values* **do**

if *value* does not conflict with any variable in *vars-done* **then**

 Assign *value* to *v*

if *mc-bt(vars-left, vars-done)* **then return** *true*

endif

Experiments

- Solves n queens problems up to a *million* queens
 - greedy initialization leads to solutions in constant number of flips
 - random initialization leads to a linear number of flips
- Used in SPIKE for Hubble Space Telescope scheduling
 - ten to thirty thousand observations per year
 - preprocessing adds inferred constraints, allowing a more accurate assessment of number of conflicts
- Graph coloring
 - Densely-connected graphs are easy for min-conflicts
 - Sparsely-connected graphs are difficult for min-conflicts

Analysis

- Simplified model
 - every variable is subject to exactly c binary constraints
 - there is only a single solution to the problem
 - an incorrect value for variable v conflicts with an arbitrary value for a connected variable v' with fixed probability p
- $Pr(\text{min-conflicts makes a mistake in repairing } v)$
 $(k - 1) e^{-2(pc - d)^2 / c}$
- Probability of a mistake decreases as
 - c becomes large, d becomes small, p increases, k decreases
- More quantitatively correct predictions are made by making better assumptions about the probability of conflicts between variables